Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №2

«Методы построения моделей машинного обучения»

Вариант № 5

Выполнила:
Буйдина К.А.
группа ИУ5-63Б

Проверил:
Гапанюк Ю.Е.

Дата: 11.04.25

Подпись:

Дата:

Подпись:

Москва, 2025 г.

Задание:

Номер варианта: **5**

Методы: дерево решений и случайный лес

Номер набора данных, указанного в задаче: **5**
https://www.kaggle.com/datasets/khushikyad001/world-happiness-report

Задача №1.

Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик).

Ход выполнения:

```python
import pandas as pd
import numpy as np

data = pd.read_csv('world_happiness_report.csv')

print(data.head())

print("\nИнформация о датасете:")
print(data.info())
print("\nПропущенные значения:")
print(data.isnull().sum())
print("\nОписательная статистика:")
print(data.describe())
```

✓ [1] 54ms

```
max    2024.000000         8.000000    59980.720000        1.000000

       Healthy_Life_Expectancy      Freedom   Generosity  \
count              4000.000000  4000.000000  4000.000000
mean                 67.917605     0.502723     0.143960
std                  10.172091     0.285219     0.200088
min                  50.000000     0.000000    -0.200000
25%                  59.177500     0.260000    -0.030000
50%                  68.015000     0.500000     0.140000
75%                  76.690000     0.750000     0.310000
max                  85.000000     1.000000     0.500000

       Corruption_Perception  Unemployment_Rate  Education_Index  ...  \
count            4000.000000        4000.000000      4000.000000  ...
mean                0.498920          10.966748         0.750385  ...
std                 0.288866           5.210712         0.144819  ...
min                 0.000000           2.000000         0.500000  ...
25%                 0.240000           6.450000         0.630000  ...
50%                 0.500000          10.995000         0.750000  ...
75%                 0.742500          15.450000         0.880000  ...
```

```
numeric_cols = data.select_dtypes(include=['float64', 'int64']).columns
data[numeric_cols] = data[numeric_cols].fillna(data[numeric_cols].mean())

print(data.isnull().sum())
```

✓ [3] 14ms

```
Country                      0
Year                         0
Happiness_Score              0
GDP_per_Capita               0
Social_Support               0
Healthy_Life_Expectancy      0
Freedom                      0
Generosity                   0
Corruption_Perception        0
Unemployment_Rate            0
Education_Index              0
Population                   0
Urbanization_Rate            0
Life_Satisfaction            0
Public_Trust                 0
Mental_Health_Index          0
Income_Inequality            0
Public_Health_Expenditure    0
Climate_Index                0
Work_Life_Balance            0
```

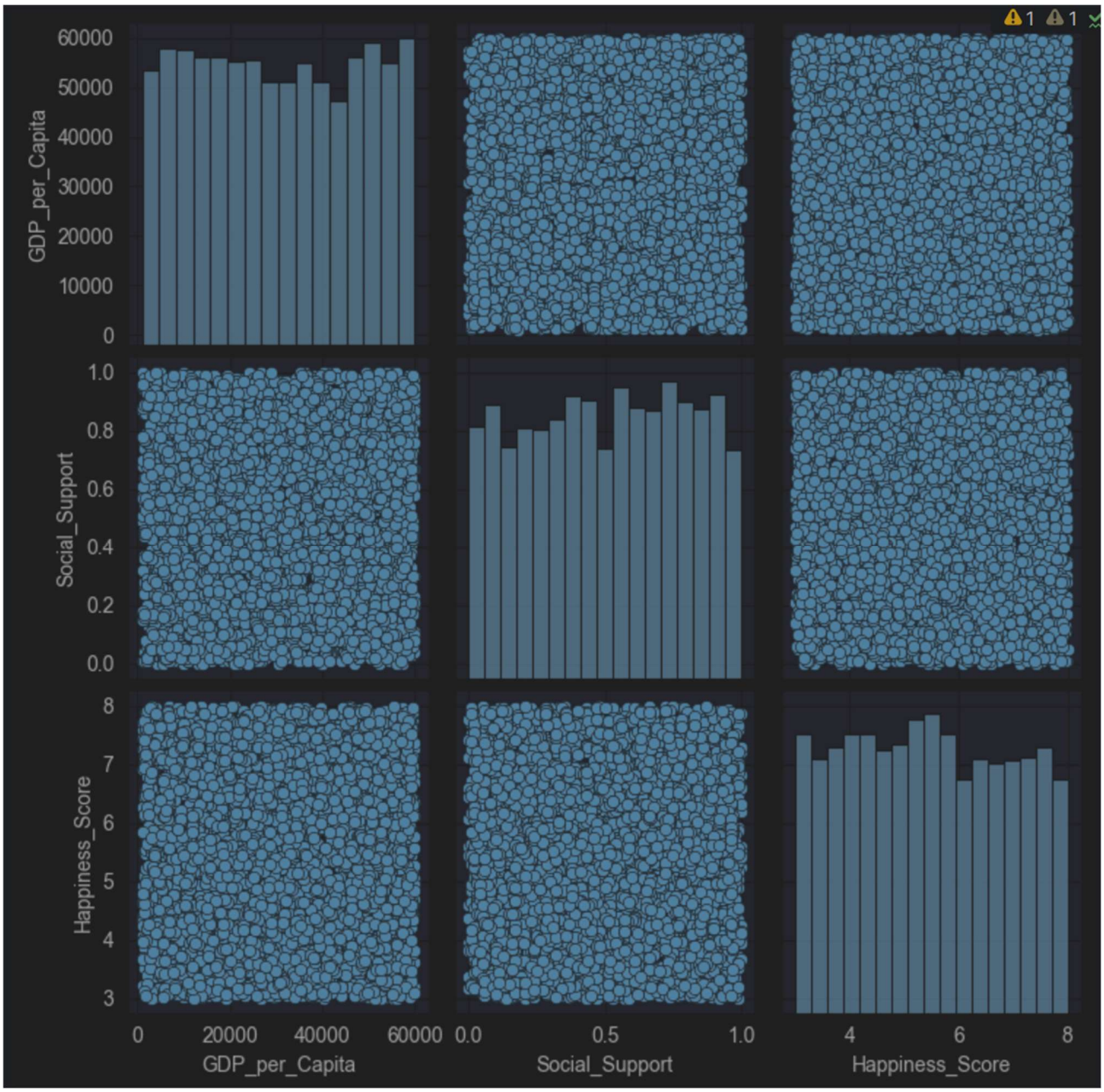
```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
data['Country_encoded'] = encoder.fit_transform(data['Country'])

data.drop('Country', axis=1, inplace=True)
```

✓ [4] 313ms

```
corr_matrix = data.corr()
print(corr_matrix['Happiness_Score'].sort_values(ascending=False))

import seaborn as sns
sns.pairplot(data[['GDP_per_Capita', 'Social_Support', 'Happiness_Score']])
```

✓ [10] 1s 13ms

```
GDP_per_Capita               0.016324
Work_Life_Balance            0.016085
Public_Trust                 0.015138
Corruption_Perception        0.015096
Life_Satisfaction            0.014920
Healthy_Life_Expectancy      0.012658
Year                         0.009508
Social_Support               0.007505
Crime_Rate                   0.003758
Internet_Access              0.003275
Education_Index              0.002752
Climate_Index               -0.001834
Public_Health_Expenditure   -0.003977
Employment_Rate             -0.004209
Generosity                  -0.004796
Urbanization_Rate           -0.004861
Country_encoded             -0.009397
Mental_Health_Index         -0.013628
Population                  -0.021352
Income_Inequality           -0.022347
```


GDP_per_Capita
Social_Support
Happiness_Score
0
20000
40000
60000
0.0
0.5
1.0
4
6
8

```
X = data.drop('Happiness_Score', axis=1)
y = data['Happiness_Score']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

✓ [5] 421ms

Построение моделей дерева решений и случайного леса

```
from sklearn.tree import DecisionTreeRegressor

tree_model = DecisionTreeRegressor(max_depth=5, random_state=42)
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)
```

✓ [6] 578ms

```
from sklearn.ensemble import RandomForestRegressor

forest_model = RandomForestRegressor(n_estimators=100, random_state=42)
forest_model.fit(X_train, y_train)
y_pred_forest = forest_model.predict(X_test)
```

✓ [7] 4s 634ms

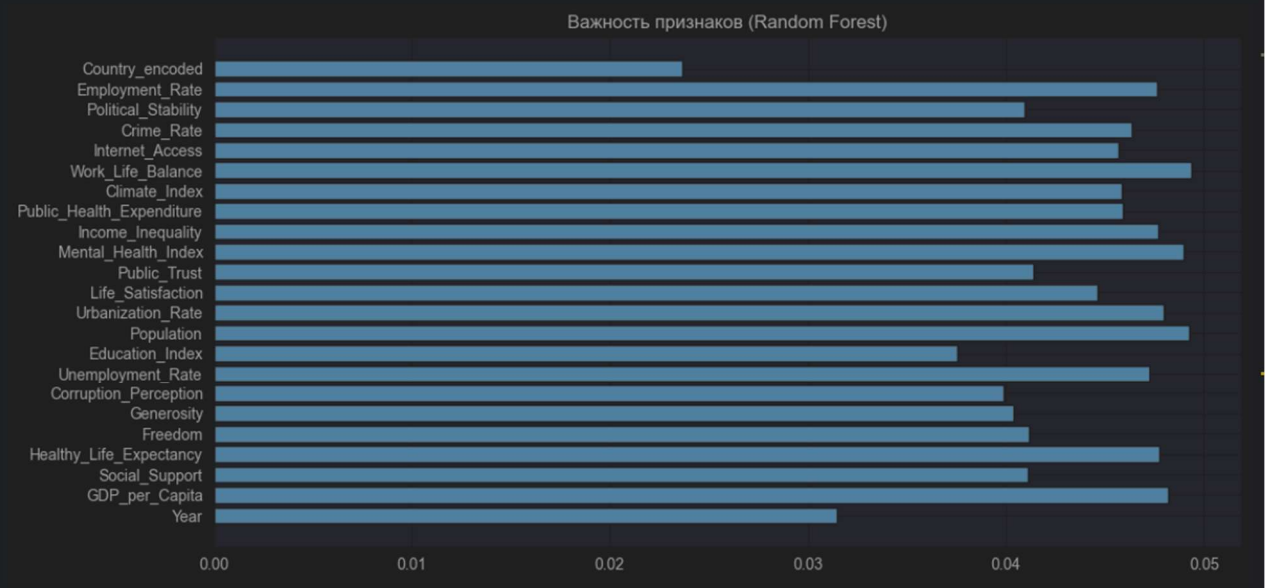
```
from sklearn.metrics import mean_absolute_error, r2_score

print("Decision Tree:")
print(f"MAE: {mean_absolute_error(y_test, y_pred_tree):.3f}")

print("\nRandom Forest:")
print(f"MAE: {mean_absolute_error(y_test, y_pred_forest):.3f}")
```

✓ [19] < 10 ms

```
Decision Tree:
MAE: 1.291

Random Forest:
MAE: 1.262
```

```python
import matplotlib.pyplot as plt

feature_importances = forest_model.feature_importances_
features = X.columns

plt.figure(figsize=(12, 6))
plt.barh(features, feature_importances)
plt.title("Важность признаков (Random Forest)")
plt.show()
```

✓ [9] 250ms



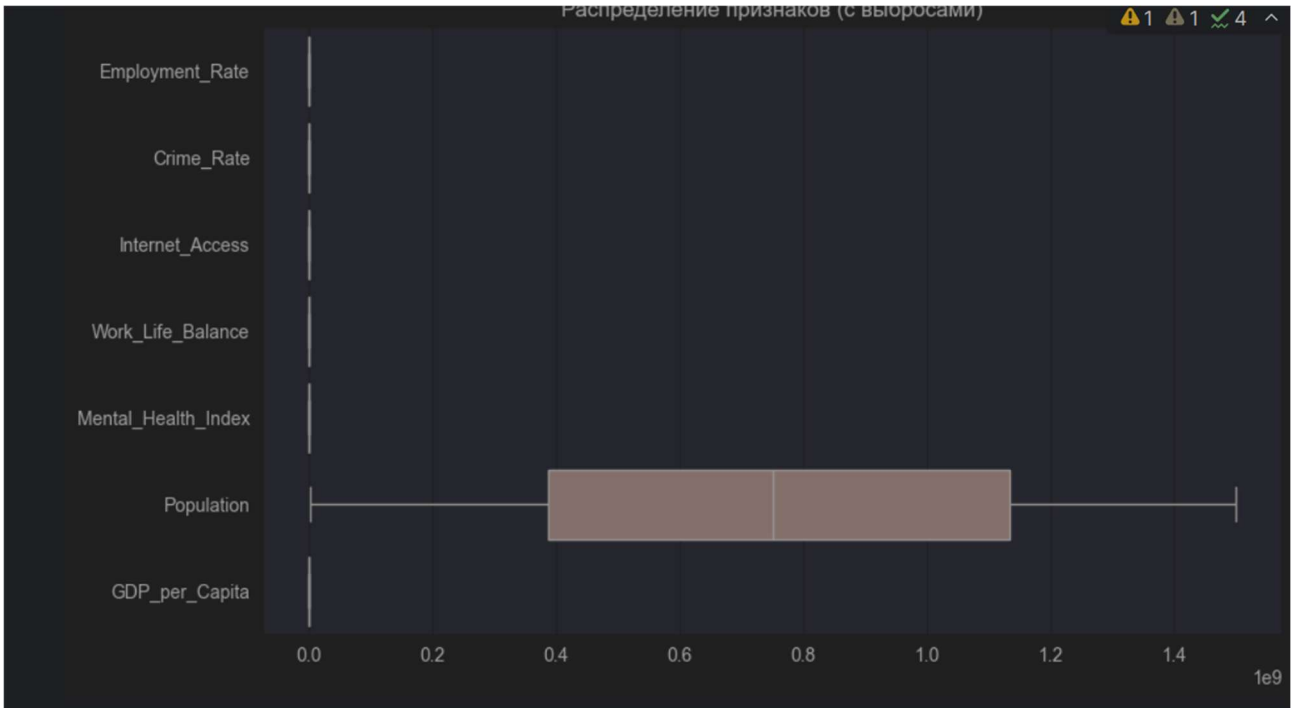
Теперь можно построить более точные модели, используя только наиболее значимые признаки

```python
selected_features = ['Employment_Rate', 'Crime_Rate', 'Internet_Access', 'Work_Life_Balance', 'Mental_Health_Index',
 'Population', 'GDP_per_Capita']
X = data[selected_features]
y = data['Happiness_Score']
```

✓ [12] < 10 ms

```python
plt.figure(figsize=(10, 6))
sns.boxplot(data=X[selected_features], orient='h')
plt.title("Распределение признаков (с выбросами)")
plt.show()
```

✓ [13] 214ms




```
Q1 = X.quantile(0.25)
Q3 = X.quantile(0.75)
IQR = Q3 - Q1

# Фильтрация выбросов
X_filtered = X[~((X < (Q1 - 1.5 * IQR)) | (X > (Q3 + 1.5 * IQR))).any(axis=1)]
y_filtered = y[X_filtered.index]
```

✓ [14] 10ms

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor

X_train, X_test, y_train, y_test = train_test_split(X_filtered, y_filtered, test_size=0.3, random_state=42)

# Дерево решений
tree_model = DecisionTreeRegressor(max_depth=3, random_state=42)
tree_model.fit(X_train, y_train)

# Случайный лес
forest_model = RandomForestRegressor(n_estimators=100, max_depth=5, random_state=42)
forest_model.fit(X_train, y_train)
```

```python
from sklearn.metrics import (
    mean_absolute_error,
    mean_squared_error,
    median_absolute_error,
    explained_variance_score,
    max_error
)

def evaluate_model(y_true, y_pred, model_name):
    print(f"\n{model_name} Metrics:")
    print(f"- MAE: {mean_absolute_error(y_true, y_pred):.3f}")
    print(f"- MSE: {mean_squared_error(y_true, y_pred):.3f}")
    print(f"- RMSE: {np.sqrt(mean_squared_error(y_true, y_pred)):.3f}")
    print(f"- MedAE: {median_absolute_error(y_true, y_pred):.3f}")
    print(f"- Explained Variance: {explained_variance_score(y_true, y_pred):.3f}")
    print(f"- Max Error: {max_error(y_true, y_pred):.3f}")

# Для дерева
evaluate_model(y_test, y_pred_tree, "Decision Tree")

# Для леса
evaluate_model(y_test, y_pred_forest, "Random Forest")
```

✓ [18] 10ms

```
 Decision Tree Metrics:
 - MAE: 1.291
 - MSE: 2.277
 - RMSE: 1.509
 - MedAE: 1.262
 - Explained Variance: -0.105
 - Max Error: 4.260

 Random Forest Metrics:
 - MAE: 1.262
 - MSE: 2.123
 - RMSE: 1.457
 - MedAE: 1.242
 - Explained Variance: -0.031
 - Max Error: 2.793
```